

FEATURE ARTICLE

Steven Kubis

Using Spreadsheets to Simulate Digital Filters

a

s a project for an independent study class, another student and I were implementing

an IIR digital filter using a 68HC11 microcontroller. We used MATLAB to design a second-order, low-pass IIR filter. MATLAB generated the transfer function that we implemented using the 68HC11. When we tested the filter, we found the output was sporadic.

Obviously, something was wrong with the filter, but we didn't know whether the problem was in the design or the implementation. To successfully troubleshoot the filter, we first had to verify that the design was correct. If we could do this, then we knew our problem was in the implementation, not the filter design.

Because it was a student project, our method had to be inexpensive. We decided to use a spreadsheet to simulate and verify the filter design.

IMPLEMENTING THE FILTER SIMULATION

We used Microsoft Excel for the Macintosh to implement the simulation. Any standard spreadsheet can be used. However, to be most useful, it's best if the spreadsheet can plot graphs directly on the worksheet (see Figure 1). To understand how the simulation works, let's first review digital filters.

Digital filters sample an input signal and calculate the output value

at specific, constant time intervals. The time between these intervals is the period. The sampling frequency of the filter is the reciprocal of the period. The characteristics of digital filters are based on the sampling frequency.

Spreadsheets work well for digital-filter simulation because instead of being periodic in time, they're periodic in position. Each row in the spreadsheet can represent one sample of the input signal and the resulting calculated output signal.

PARTS OF THE SPREADSHEET

The spreadsheet is composed of six parts:

- **sample column**—serves as the timebase for the simulated filter and is used as a basis for the input and output plot. The input signal for the simulated filter is derived using the sample column. For most simulations, 100 data points are adequate.
- **input column**—produces the simulated-input signal for the filter. The values in this column are computed based on the sample column and the values of the input signal frequency and sampling frequency. This is described in detail in the next section.
- **filtered column**—contains the output of the simulated filter based on the input signal and the filter coefficients.
- **signal and filter characteristics**—specify the input signal frequency and the sampling frequency of the filter. They also specify the input-signal magnitude and any offset.
- **filter coefficients**—come from the discrete-time transfer function used to describe the digital filter. These coefficients are used to calculate the values in the filtered column.
- **input and output plot**—enables the input signal and filtered output signal to be viewed. The plot is produced by plotting the values in the input and filtered columns versus the sample column.

PRODUCING THE INPUT SIGNAL

The input signal is the most difficult part of the spreadsheet to

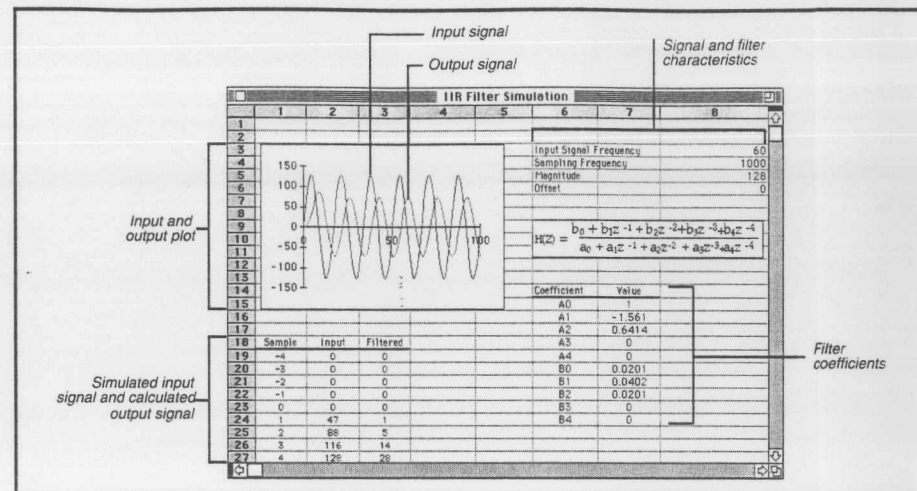


Figure 1—Microsoft Excel running on a Macintosh works well for filter simulation because it can display graphs of the data on the same screen as the spreadsheet itself.

implement. The input signal should be periodic and have characteristics of standard input signals (standard input signals include sine, square, and triangular waves). It also must demonstrate the correct relationship between the frequency of the input signal and the sampling frequency. Two basic calculations produce these results.

To be periodic, the number of samples in one period of the input signal must be calculated. This value is determined by the ratio of the sampling frequency and the input frequency.

$$\text{Samples Per Period} = \frac{\text{Sampling Frequency}}{\text{Input Frequency}}$$

One cycle of the input signal must be completed in this number of samples.

To correctly produce the input signal, the current position in the current period of the input signal must also be determined, as shown in Figure 2. The position in the period (PP) is calculated by:

$$PP = \text{Current Sample \%} \left(\frac{\text{Sampling Frequency}}{\text{Input Frequency}} \right)$$

where % is the modulus operator. Based on these calculations, the different input signals are produced (descriptions of some standard input

signals follow). Be sure to use absolute references for the input and sampling frequency, magnitude, and offset. Use a relative reference for the current sample.

The sine wave is simulated using the SIN() function found in standard spreadsheets. The signal is scaled based on the magnitude, rounded to the nearest integer, and then offset:

$$\text{Round} \left(\text{SIN} \left(\frac{PP \times 2\pi \times \text{Input Frequency}}{\text{Sampling Frequency}} \right) \times \text{Magnitude} \right) + \text{Offset}$$

The square wave is simulated using the IF() operation and COS() function found in standard spreadsheets. When the output of the COS() function is positive, the signal is the magnitude plus the offset. When the COS() function is negative, the signal is the offset less the magnitude:

$$\text{IF} \cos \left(\frac{PP \times 2\pi \times \text{Input Frequency}}{\text{Sampling Frequency}} \right) > 0 \text{ THEN} \\ \text{Magnitude} + \text{Offset} \\ \text{ELSE} \\ \text{Offset} - \text{Magnitude}$$

The triangle wave is simulated by two parametric equations. The signal starts at the positive magnitude and decreases to the negative magnitude by the middle of the period. In the second half of the period, the signal starts at the negative magnitude and increases

to the positive magnitude. The IF() operation determines which half of the period is currently being calculated (see Figure 3).

The output of the filter is calculated from the difference equation for the filter being simulated. The form of the difference equation varies depending on the type of filter. When entering the formula to compute the output, you should use absolute references to the filter coefficients. Using this method, you have to write only one formula, which can be copied to all rows in the output column.

USING THE SPREADSHEET

This spreadsheet can be used to simulate IIR filters, FIR filters, and

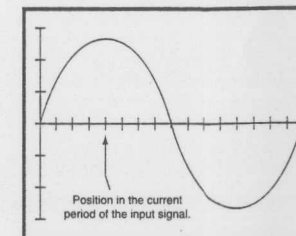


Figure 2—The current position in the current period of the input signal must be determined for the input signal to be periodic. In this illustration, the value of the input signal for the fourth of sixteen positions is calculated.

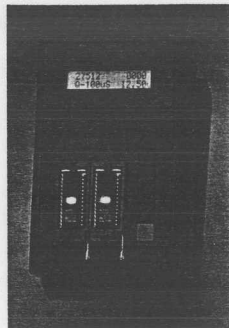
010
111
010
**DATA
Genie™**

Data Genie offers a full line of test & measurement equipment that's innovative, reliable and very affordable. The "Express Series" of stand-alone, non-PC based testers are the ultimate in portability when running from either battery or AC power. Data Genie products will be setting the standards for quality on the bench or in the field for years to come.



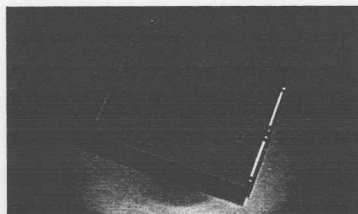
HT-28 Express
DIGITAL IC TESTER

The HT-28 is a very convenient way of testing Logic IC's and DRAMs. Tests most TTL 74, CMOS 40/45 and DRAMs 4164-414000, 44164-441000. It can also identify unknown IC numbers on TTL 74 and CMOS 40/45 series with the "Auto-Search" feature.
\$189.95



HT-14 Express
EPROM PROGRAMMER

The HT-14 is one-to-one EPROM writer with a super fast programming speed that supports devices from 2732B to 27080, with eight selectable programming algorithms and six programming power (VPP) selections.
\$289.95



P-300

PC INTERFACE CARD PROTECTOR

The Data Genie P-300 is a useful device that allows you to quickly install add-on cards or to test prototype circuits for your PC externally. Without having to turn off your computer to install an add-on card, the P-300 maintains complete protection for your motherboard via the built-in current limit fuses.
\$349.95

MING
Microsystems
Division of MING & P, INC.
17921 Rowland Street
City of Industry, CA 91748
TEL: (818) 912-7756
FAX: (818) 912-9598

Call for a dealer near you.
1-800-473-6606

Data Genie products are backed by a full 1 year limited factory warranty.

©1994 MING Microsystems. All rights reserved. Data Genie logo is a registered trademark of MING Microsystems.

CCDG01

analog filters that have been converted to discrete-time form. I suggest you make a template for each type of filter and input, then begin experimenting. The description of three filters follow. Use these filter designs to test your templates.

The following equation is the generalized transfer function for an IIR filter of order q .

$$H(Z) = \frac{Y(Z)}{X(Z)} = \frac{b_0 + b_1 Z^{-1} + \dots + b_q Z^{-q}}{a_0 + a_1 Z^{-1} + \dots + a_q Z^{-q}}$$

In Figure 4, we see what converting this to a difference equation yields. The value of the difference equation is computed to produce the filter output. When calculating the difference equation, it's a good idea to use as many coefficients as the highest-order filter needs. When simulating lower-order filters, enter zeros for the higher-order coefficients.

Note that the output is calculated from the current input and previous input and output values. For the filter to be causal, you need to use zero as the input for the samples prior to sample 1. The number of these zero samples depends on the order of the filter you're simulating. Also, when you write the output formula, be sure to use relative references to the previous input and output values.

After you've created the template for an IIR filter, try simulating the filter shown in Figure 5a. This filter is a second-order, low-pass, Butterworth filter, designed using MATLAB for a sampling frequency of 1000 Hz and a cutoff frequency of 50 Hz.

The following is the generalized transfer function for an FIR filter of order q .

$$H(Z) = \frac{Y(Z)}{X(Z)} = a_0 + a_1 Z^{-1} + a_2 Z^{-2} + \dots + a_q Z^{-q}$$

Converting this to a difference equation yields:

$$y(n) = a_0 x(n) + a_1 x(n-1) + \dots + a_q x(n-q)$$

Again, it's the difference equation that's calculated to produce the output. The difference equation is much simpler for a FIR filter, but FIR filters must be of a much higher order to have the desired characteristics. This makes FIR filters less practical to

$$\text{IF } \text{PP} \geq \left(\frac{\text{Sampling Frequency}}{2 \times \text{Input Frequency}} \right) \text{ THEN} \\ \text{Magnitude} = \left[\frac{4 \times \text{Input Frequency} \times \text{Magnitude}}{\text{Sampling Frequency}} \right] \times \left(\text{Current sample \%} \left(\frac{\text{Sampling Frequency}}{2 \times \text{Input Frequency}} \right) \right) + \text{Offset} \\ \text{ELSE} \\ \text{Magnitude} = \left[\frac{4 \times \text{Input Frequency} \times \text{Magnitude}}{\text{Sampling Frequency}} \right] \times \left(\text{Current sample \%} \left(\frac{\text{Sampling Frequency}}{2 \times \text{Input Frequency}} \right) \right) + \text{Offset}$$

Figure 3—The formula to simulate a triangle-wave input signal uses an IF() operation to determine which half of the period is currently being calculated.

simulate using a spreadsheet, though it can be done.

After you've created the template for the FIR filter, try simulating the filter given in Figure 5b. This filter is a tenth-order notch filter designed with

$$y(n) = \frac{b_0 x(n) + b_1 x(n-1) + \dots + b_q x(n-q) - [a_1 y(n-1) + \dots + a_q y(n-q)]}{a_0}$$

Figure 4—A difference equation is used to calculate the output of an IIR filter of order q .

MATLAB for a sampling frequency of 1000 Hz. It has a center frequency of 125 Hz and a bandwidth of approximately 100 Hz.

Analog filters can be simulated if they have been converted to a discrete-time form. The trapezoid rule is a good conversion to use to transform an analog filter from continuous-time

to discrete-time form. The filter's transfer function is:

$$G(s) = \frac{10}{10 + 0.2533s + \frac{100000}{s}}$$

Discretized using the trapezoid rule and a 1000-Hz sampling frequency, the discrete-time form is:

$$H(Z) = \frac{1 - Z^{-2}}{56.66 - 91.322Z^{-1} + 54.622Z^{-2}}$$

You can use the template you created for the IIR filter to verify the characteristics of this analog filter.

EXPERIMENTATION

After your templates work correctly, you can begin to experiment with the items listed below:

- step response and settling time—make a template with a unit step input to view the step response and estimated settling time.
- aliasing—watch for aliasing to occur when the sampling frequency is too low for the input signal.
- output signal quality—note that the quality of the output signal degrades as the input signal approaches the sampling frequency.

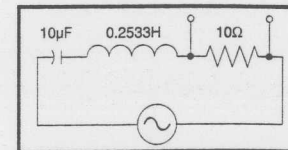


Figure 6—One example of an analog filter that can be simulated is a band-pass filter with a resonant center frequency of 100 Hz. The output is across the resistor.

- unstable filters—add an extra term to the denominator of the example IIR filter. This will add an unstable pole to the filter, causing the output to "explode." (Fortunately, in the simulation, this just means extremely large output values, not physical damage.)
- noise—use the RAND() function in your spreadsheet to add simulated noise to the input signal.

The only limits are the spreadsheet's capabilities and your own creativity.

CONCLUSION

Basic digital filters can be simulated using a spreadsheet, a tool most people already have. Once you've created a set of templates, the simulations are easy to use, flexible, accurate, and best of all inexpensive.

Getting back to my original problem, the spreadsheet simulation showed that the filter design was correct. The problem was in the hardware. After some debugging, the filter worked like the design. It was a beneficial problem, though. Now my colleague and I know that spreadsheets aren't just a financial tool.

Steven Kubis is currently a senior technical writer with Great Plains Software in Fargo, North Dakota. He graduated in 1993 with a BSEE degree from North Dakota State University. He may be reached at skubis@cogs.gps.com.

REFERENCES

- R. E. Ziemer, W. H. Tranter, and D. R. Fannin, *Signals and Systems: Continuous and Discrete*, New York: Macmillan Publishing Company, 1989.
- The Student Edition of MATLAB Reference Manual, Englewood Cliffs: Prentice Hall, 1992.

I R S

- 407 Very Useful
- 408 Moderately Useful
- 409 Not Useful